Bericht Todesstern U2

Charline Waldrich, Robert Ullmann, Julian Dobrot

13. november 2015

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Strahl

Die Klasse Ray representiert einen Strahl. Im konstruktor dieser Klasse soll der Ursprung und die Richtung des Strahls übergeben werden.

1.2 Kameras

Es sollen drei Kameraklassen implementiert werden.

1.2.1 Camera

Diese Klasse ist die abstrakte Basisklasse. sie speichert die drei Vektoren e,g,t welche im Konstruktor übergeben werden. Außerdem wird die

Kameratransformation der vektoren u,v,w auch in ihrem Konstruktor berechnet.

1.2.2 OrthographicCamera

Soll eine orthogrsphische Kamera wiederspiegeln und hat als weiteren Parameter den Skalierungsfaktor der bildebene.

1.2.3 PespectiveCamera

Eine perspektivische Kamera mit dem Öffnungswinkel als weiterem Parameter

1.3 Farben

Die Klasse Farben soll Farben im RGB-Farbraum representieren. Dabei sollen die einzelnen Komponenten einen Wert von 0 und 1 annehmen. Farben sollen addiert, subtrahiert und multipliziert werden. Desweiteren sollen Farben mit einem skalar mutlipliziert werden können.

1.4 Geometrien

Alle Geometrien erben von einer abstrakten Basisklasse. Diese hat nur ein Attribut nämlich die Farbe.

1.4.1 Hit

Die Klasse Hit wird instanziert, wenn ein Strahl eine Geometrie in der Welt Schneidet.

1.4.2 Plane

Die Klasse Plane soll eine Implementierung einer Ebenel darstellen.

1.4.3 Sphere

Die Klasse Sphere soll eine Implementierung einer Kugel darstellen.

1.4.4 Triangle

Die Klasse Triangle soll eine Implementierung eines Dreicks darstellen.

1.4.5 AxisAlignedBox

Die Klasse Sphere soll eine Implementierung einer Box darstellen.

1.5 Welt

In der Welt sollen die darzustellenden Objekte in einer Scene dargestellt werden

1.6 Raytracer

Hier soll über jeden pixel des Bildes iteriert werden. Werden Schnittpunkte mit Geometrien gefunden, erhält der Pixel dessen Farbe.

2 Lösungsstrategien

Allgemein wurden als erster Lösungsschritt alle aus der Aufgabenstllung hervorgehenden Klassendiagramme komplett in das Projekt integriert.

2.1 Strahl

Der Strahl (Ray) kann mit der Methode ,,at" den Punkt berechnen, welchen er trifft wenn er die als Parameter übergebenen Länge ,,t" (als

double Wert), den Richtungsvektor "entlang geht". Hierfür wird die eine einfache Rechnung angewandt. In der Methode ,,tOf" wird diese Rechnung umgedreht. Ein gesuchter Punkt wird als Parameter übergeben und berechnet dann den Faktor ,,t" (als double Wert) welcher mit dem Richtungsvektor multipliziert werden müsste um zu gesuchtem Punkt zu gelangen.

2.2 Kameras

2.2.1 Camera

Die Kamera führt die Transformation der Position, Richtung und Rotation durch.

2.2.2 OrthographicCamera

Bei der Orhographischen Kamera müssen alle erzeugten Strahlen parallel sein und in die gleiche Richtung zeigen. In der abstrakten Klasse Camera wurde mit den aus der Vorlesung vermittelten Formeln die Kameratransformation vorgenommen.

2.2.3 OrthographicCamera

In der orthographischen Kamera wurden die folgenden Formeln umgesetzt:

$\vec{d} = -\vec{w}$

$\vec{o} = \vec{e} + a * s\frac{px-\frac{w-1}{2}}{w-1}\vec{u} + s\frac{py-\frac{h-1}{2}}{h-1}\vec{v}$

2.2.4 PespectiveCamera

In der perspektivischen Kamera wurden die folgenden Formel umgesetzt:

$\vec{o} = \vec{e}$

$\vec{r} = -\vec{w} * \frac{\frac{h}{2}}{\tan\alpha} + (px - \frac{w-1}{2})\vec{u} + (py - \frac{h-1}{2})\vec{v}$

$\vec{d} = \frac{\vec{r}}{|\vec{r}|}$

Der Winkel in $\vec{r}$ wurde halbiert, um die aus der aufgabenstellung gewünschten Bilder zu erhalten.

2.3 Farben

Die Klasse der Farbe, repräsentiert eine Farbe, welche ein neues Objekt vom Typ Color erzeugen kann.

2.4 Geometrien

2.4.1 Hit

Die Klasse Hit konnte rein aus der UML darstellung übernommen werden. sie implementiert keine Methoden.

2.4.2 Plane

Um die hit Methode in dieser Klasse zu implementieren wurde die implizite ebenen Formel aus der Vorlesung angewendet.
$\Rightarrow t = \frac{(\vec{a}-\vec{o})\bullet\vec{n}}{\vec{d}\bullet\vec{n}}$

2.4.3 Sphere

Als Lösungsansatz für diese Geometrie dienten die aus der Vorlesung erarbeite Formel zur berechnung der Schnittpunkte verwendet.

$t = \frac{-b \pm \sqrt[2]{b^2 - 4ac}}{2a}$

Für die berechnung der Anzahl der Schnittpunkte mit der Sphäre wurde die folgende Formel herangezohgen:

$d = b^2 - 4ac$

Wobei der Strahl bei $d < 0$ keinen Schnittpunkt mit der Kugel aufweist. Bei $d = 0$ genau einen und bei $d > 0$ zwei Schnittpunkte.
Die einzelnen Komponennten dieser formeln setzen sich wie folgt zusammen:

$a = d^2$

$b = d * [2(o - c)]$

$c = (o - c) * (o - c) - r^2$

Diese Überlegungen wurden dementsprechend programmtechnisch umgesetzt, um eine gewünschte implementierung der hit-Methode zu erzielen.

2.4.4 Triangle

Die Triangle besteht aus 3 Punkten, welche die Baryzentrischen Koordinaten bilden. Um den Schnittpunkt zu berechnen, ist es notwenig mit Hilfe der 3x3 Matrix ein lineares Gleichungssystem zu erstellen und es mit der Cramerschen Regel, unter Verwendung der Determinanten aus der 3x3 Matrix zu lösen. Für die lösung dieser Geometrie wurden die Formeln aus der Vorlesung angewendet.

2.4.5 AxisAlignedBox

Die hit Methode wurde wie folgt überschrieben.Die jeweils drei Ebenen die die linke nahe Ecke und die rechte entfernte Ecke aufspannen wurden in Form von Planes instanziiert und mit ihren ensprechenen Normalenparametern versehen. Diese sechs Ebenen wurden in eine Liste gespeichert die dann in einer Schleife durchlaufen wird. Es wird im folgenden mit den Formeln aus dem unterricht getestet ob Strahlen die Box treffen oder nicht

2.5 Welt

2.6 Raytracer

3 Implementierungen

3.1 Strahl

Die ,,Ray" Klasse initialisiert einen Strahl mit gegebenem Ausgangspunkt und Richtungsvektor. Sie implementiert zwei Methoden namens ,,at" und ,,tOf". Außerdem wurden die von der Oberklasse Object geerbten Methoden hashCode und eqals so überschrieben, dass sie zum Testen nützlich sind.

Der Strahl (Ray) kann mit der Methode ,,at" den Punkt berechnen, welchen er trifft wenn er die als Parameter übergebenen Länge ,,t" (als double Wert), den Richtungsvektor "entlang geht". Hierfür wird die eine einfache Rechnung angewandt. In der Methode ,,tOf" wird diese Rechnung umgedreht. Ein gesuchter Punkt wird als Parameter übergeben und berechnet dann den Faktor ,,t" (als double Wert) welcher mit dem Richtungsvektor multipliziert werden müsste um zu gesuchtem Punkt zu gelangen.

3.2 Kameras

3.2.1 Camera

Die abstrakte Camera klasse berechnet die Kameratransformation.

3.2.2 OrthographicCamera

Die Klasse OrthographicCamera bekommt 2 Vektoren und einen Punkt übergeben, zusätzllch noch den Öffnungswinkel. In der öffentlichen Methode rayfor, wird der sichtbare Bereich in Höhe und Breite festgelegt und mit Hilfe der Formel der Orthographischen Kamera der Strahl zurückgegeben, welcher für die Projektion notwendig ist.

3.2.3 PespectiveCamera

Die Klasse PerspectiveCmera bekommt einen Punkt, zwei Vektoren und einen Öffnungswinkel übergeben.

3.3 Farben

Die übergebenen Variablen vom Typ double stehen für die Farben Rot, Gelb , Blau und werden im Konstruktur dem Objekt selbst übergeben.

3.4 Geometrien

3.4.1 Hit

Das Hit Objekt bekommt einen double Wert ,,t", einen Strahl ,,ray" und eine geometrische Figur übergeben und zugewiesen. Außerdem werden in der Klasse die von der Oberklasse Object vererbten Methoden ,,toString", ,,hashCode" und ,,equals" überschrieben.

3.4.2 Plane

Die Klasse Plane implementiert die Geometrie der Ebene. Sie hat die Öffentlichen Attribute a und n, wobei a ein Punkt ist der garantiert auf der Ebene liegt und n eine Normale. Die Klasse hat die öffentliche Methode hit die Schnittpunkte mit Strahlen berechnet.

3.4.3 Sphere

Die Klasse Sphere implementiert die Geometrie einer Kugel. Sie bekommt im Konstruktor einen Punkt und einen Radius übergeben. In der Methode Hit werden Schnittpunkte von Strahlen mit der Geometrie berechnet.

3.4.4 Triangle

Zu Beginn werden die Übergebenen Variablen auf Plausibilität geprüft, danach wird das linierae GLeichungssystem mithilfe einer Matrix aus der Matrix-Vektor-Bibliothek erstellt. Im folgenden wird druch das tauschen der Zeilen einer der Faktoren eliminiert. Am Ende kann mit Hilfe der Cramerschen Regel und den Determinanten der Matrizen β , γ und t bestimmt werden.

3.4.5 AxisAlignedBox

Die Klasse AxisAlignedBox hat zwei öffentliche attribute lbf und run, die zum einen die linke nahe Ecke und zum anderen die rechte entfernte Ecke bestimmen. Im Konstruktor werden diese beiden Punkte übergeben als auch eine Farbe. die Klasse implementiert die Methode Hit die Schnittpunkte mit Strahlen berechnet und die Methode compareDis die überprüft ob hits wirklich auf der Box liegen.

3.4.6 Plane

Die Klasse Plane beschreibt eine Ebene. Diese wird genau definiert durch die übergebenen Parameter; einen auf der Ebene liegenden Punkt, einem Normalen-Vektor welcher die Ausrichtung (bzw. Neigung) der Ebene definiert, und einem Color Objekt welche die Farbe der Ebene hält.
Die von der Oberklasse Object geerbten Methoden ,,toString", ,,hashCode" und ,,equals" wurden sinnvoll überschrieben.
Des weiteren wird die Methode ,,hit" von der abstrakten Oberklasse Geometry vererbt und auf die Ebene angepasst. Diese bekommt einen Strahl übergeben welcher nicht null sein darf und gibt ein Hit Objekt zurück. Danach wird zunächst getestet ob das Skalarprodukt des Strahls und des Normalenvektor gleich Null ist. Ist dies der Fall sind Ebene und Strahl paralell und ,,null" wird zurück gegeben.
Ist dies nicht der Fall, wird für den Strahl und die Ebene die Entfernung berechnet.
Ist diese kleiner Null bedeutet das, dass das Objekt vor dem Ortsvektor des Strahls (und somit vor unserer Bildschirmfläche) liegt und es wird wieder ein ,,null" Objekt zurück geliefert.
Ist der berechnete double-Wert größer 0 wird ein Hit Objekt mit dem berechneten ,,t"-Wert, dem der Methode übergebenen Ray ,,r" und der im Konstruktor zugewiesenen Farbe zurück gegeben (siehe Abb.

3.4.6).
Außerdem wird in unserer Color Klasse der RGB-Wert berechnet, welcher für das färben des writableImage Objektes nicht brauchbar ist. Da der pixelWriter nur Color Objet aus dem javaFX Paket erkennt, müssen die einzeln berechneten Werte an ein neues Color Objekt dessen übergeben werden.

3.5 Welt

Die Klasse ,,World" initialisiert ein Fenster mit übergebener Hintergrundfarbe. Diese soll zukünftig vom Nutzer ausgewählt werden. Sie beinhaltet eine Array Liste in welcher nur Objekte des Typs ,,Geometry" gespeichert werden können.
Des weiteren gibt es eine Methode namens ,,hit" welche einen Strahl (Ray) übergeben bekommt und eine Farbe vom Typ Color zurück gibt. Zunächst wird ein Hit Objekt ,,hit0" der mit dem null-Wert initialisiert.
In der for-Schleife wird für jedes Objekt in der Liste ein weiteres Hit Objekt mit dem übergebenen Strahl gespeichert. Hält das ,,hit1" Objekt einen Wert welcher ungleich null ist und liegt vor den anderen Objekten (hat also ein kleineren ,,t"-Wert, als die Objekte welche davor in hit0 gespeichert wurden), so wird der außen liegenden Variable ,,hit0" dieses Hit Objekt zugewiesen. Sollte nach dem Durchgang der kompletten Schleife der Wert von ,,hit0" immer noch null sein, so wird die Hintergrundfarbe der Welt zurück gegeben. Wenn nicht so wird immer die Farbe des am nächstliegendsten Objekt zurück gegeben.

3.6 Raytracer

4 Besondere Probleme oder Schwierigkeiten bei der Bearbeitung

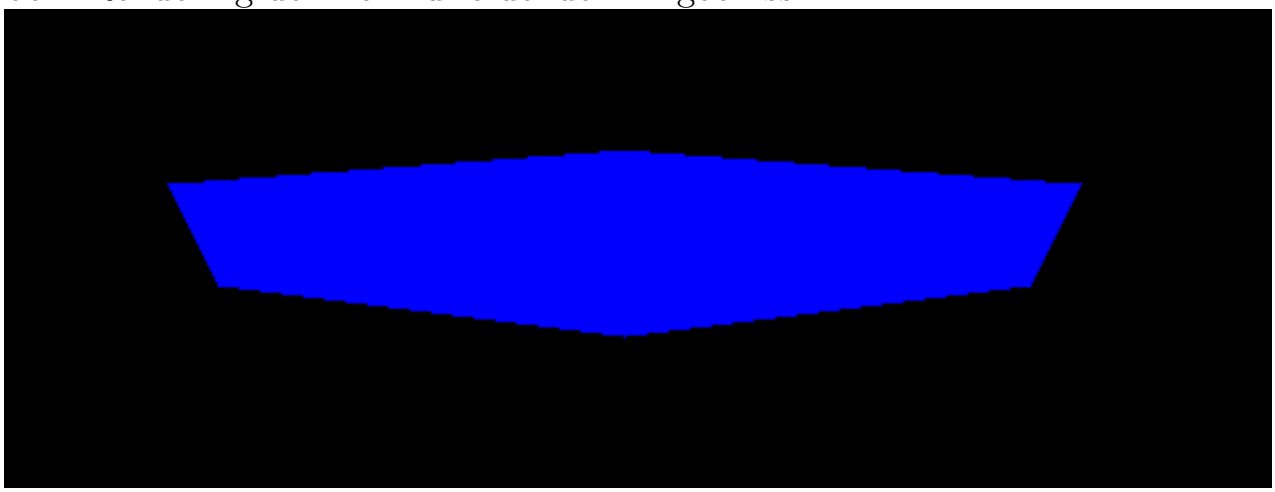
4.1 Strahl

Keine besonderen Probleme oder Schwierigkeiten.

4.2 Kameras

4.2.1 Camera

Ein Fehler in der Camera klasse führte zu einer überarbeitung der Kameratransformation, da Koordinaten falsch berechnet wurden, kam es bem Rendering der Box zu foldendem Ergebniss.



4.2.2 OrthographicCamera

Problem bei der Implementierung ist die späte Feststellung, ob die Kamera richtig Implementiert wurde, da ein Debuggen nur bedingt

möglich ist, da sehr viele Straheln (Ray) erzeugt werden. Keine besonderen Probleme oder Schwierigkeiten.

4.2.3 PespectiveCamera

Keine besonderen Probleme oder Schwierigkeiten.

4.3 Farben

Keine besonderen Probleme oder Schwierigkeiten.

4.4 Geometrien

4.4.1 Hit

Keine besonderen Probleme oder Schwierigkeiten.

4.4.2 Plane

Keine besonderen Probleme oder Schwierigkeiten.

4.4.3 Sphere

Ein Fehler in einer der Methoden der Klasse Vector3 führte zu einer längeren Fehlersuche. Durch einen Operatorschreibfehler kam es zu einer Fehlerhaften berechnung wie im folgendem Bild zu sehen ist:

4.4.4 Triangle

Große Probleme bei der Implementierung der Klasse Triangle, gab es beim Verständnis der Formel und der richtigen Implementierung in Java-Code. Auch mögliche Fehler in der Vektor-Bibliothek führten zu nicht gleich offensichtlichen Problemen.

4.4.5 AxisAlignedBox

Das Testen ob ein Hit wirklich auf der Würfelfläche liegt oder nicht bereitete einige schwierigkeiten, konnte aber mit etwas Recherche gelöst werden. Außerdem führte ein fehler in der abstrakten Camera Klasse zu falschen berechnungen der Winkel und Seitenlängen.

4.5 Welt

Keine besonderen Probleme oder Schwierigkeiten.

4.6 Raytracer

In der ImageSaver Klasse musste die ,,getColor"-Methode angepasst werden. Diese benutzt unsere persönliche Color Klasse und berechnet für jeden einzelnen Pixel dank der ,,hit" Methode der Welt, ob das erzeugte Objekt von dem Strahl der erzeugten Klasse getroffen wird. Da in der Bildverarbeitung der Ursprung des Koordinatensystems oben links und in der Mathematik unten links liegt, musste bei der Berechnung aufgepasst werden, dass der ,,rayFor" Methode welche von der ,,camera" aufgerufen wird. nicht die oben liegende Y-Koordinate übergeben wird, sondern dem quasi spiegelverkehrtem Pixel unten.

5 Zeitbedarf

Strahl	60 min
Kameras	120 min
Farben	60 min
Geometrien	720 min
Welt	60 min
Raytracer	240 min
Bericht	180 min
	1440 min

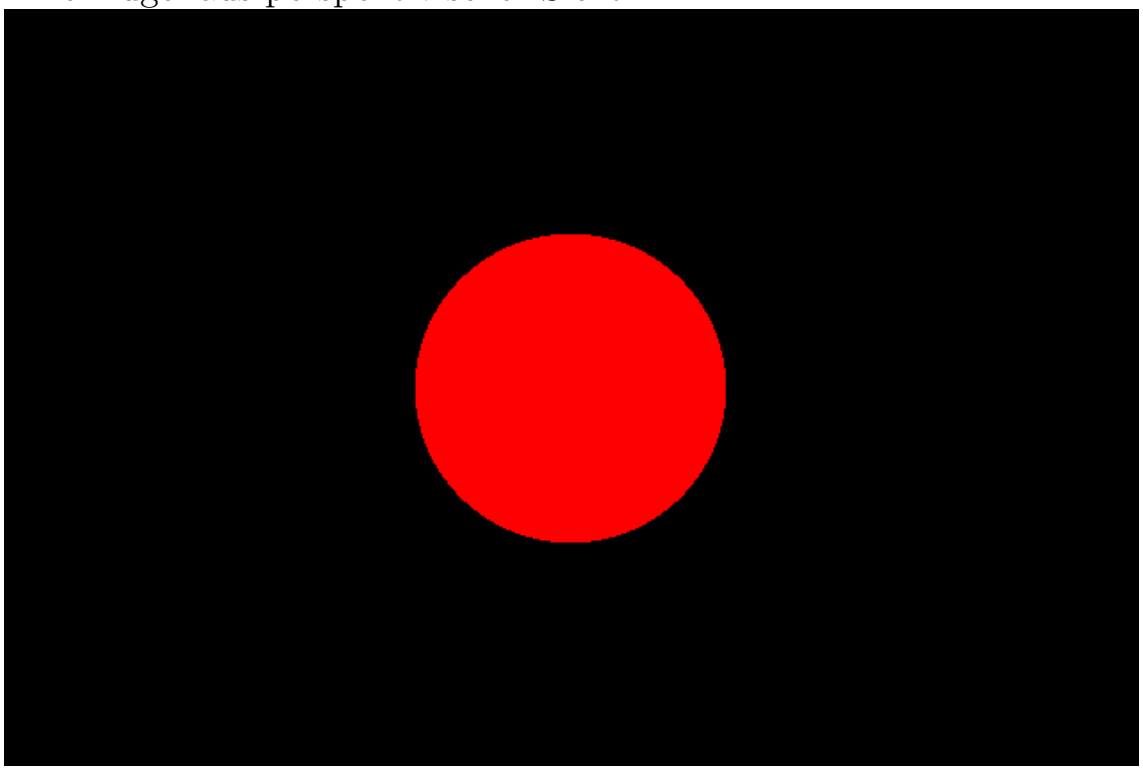
5.1 Tests

5.1.1 Abbildung 5

Eine Ebene aus perspektivischer Sicht.

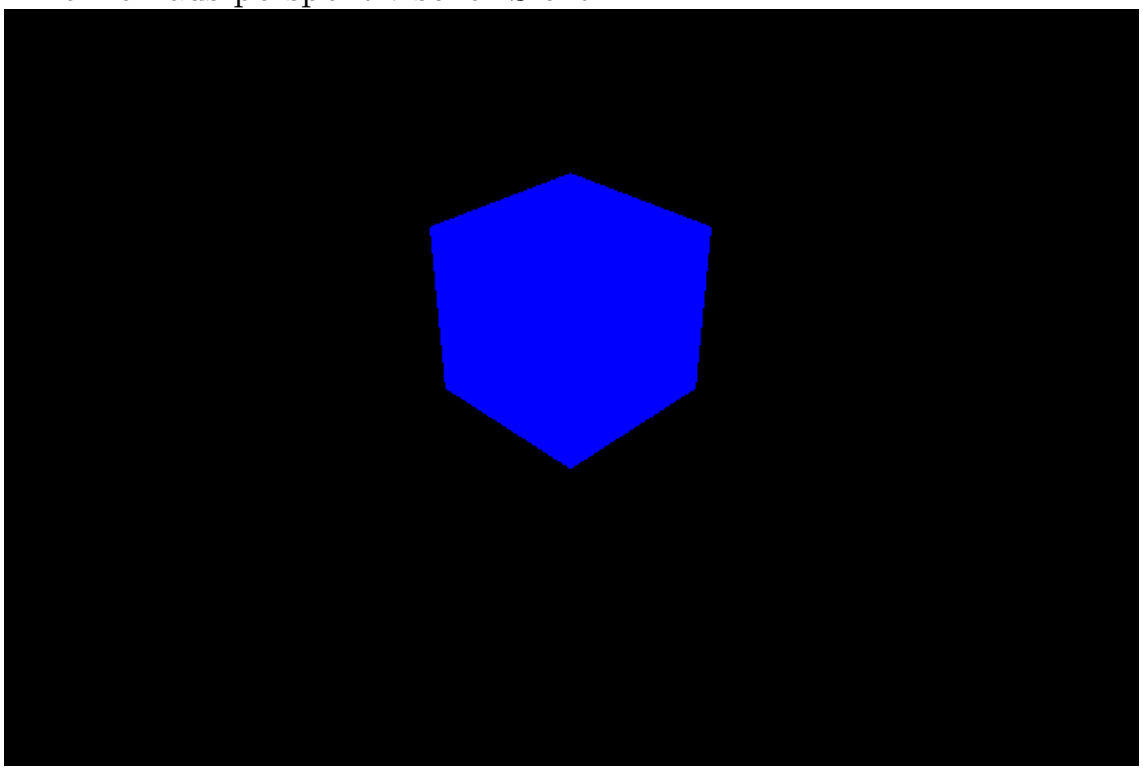
5.1.2 Abbildung 6

Eine Kugel aus perspektivischer Sicht.

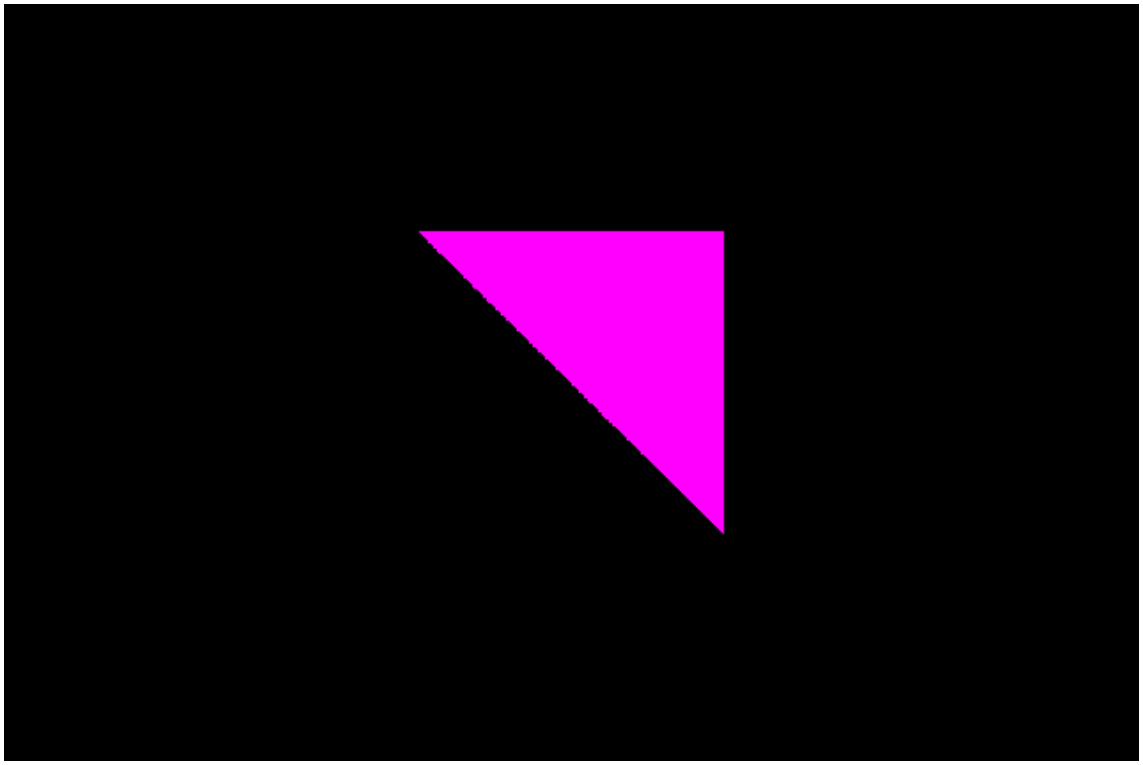


5.1.3 Abbildung 7

Eine Box aus perspektivischer Sicht.

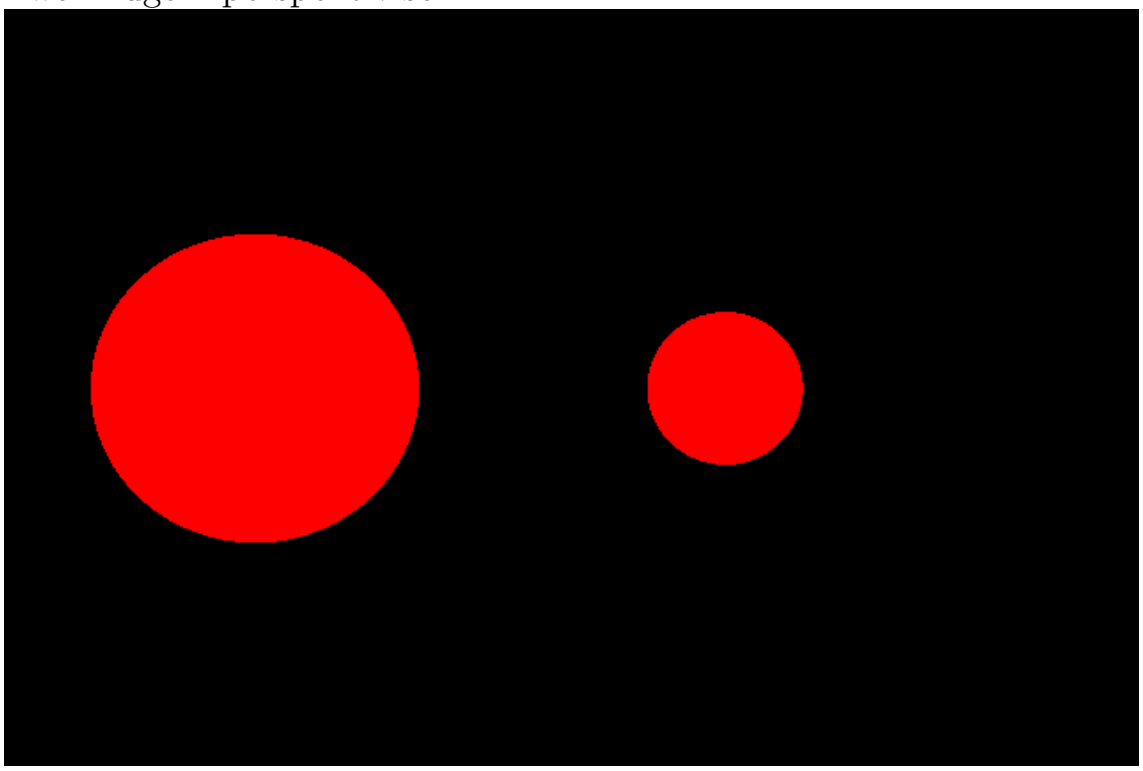


5.1.4 Abbildung 8



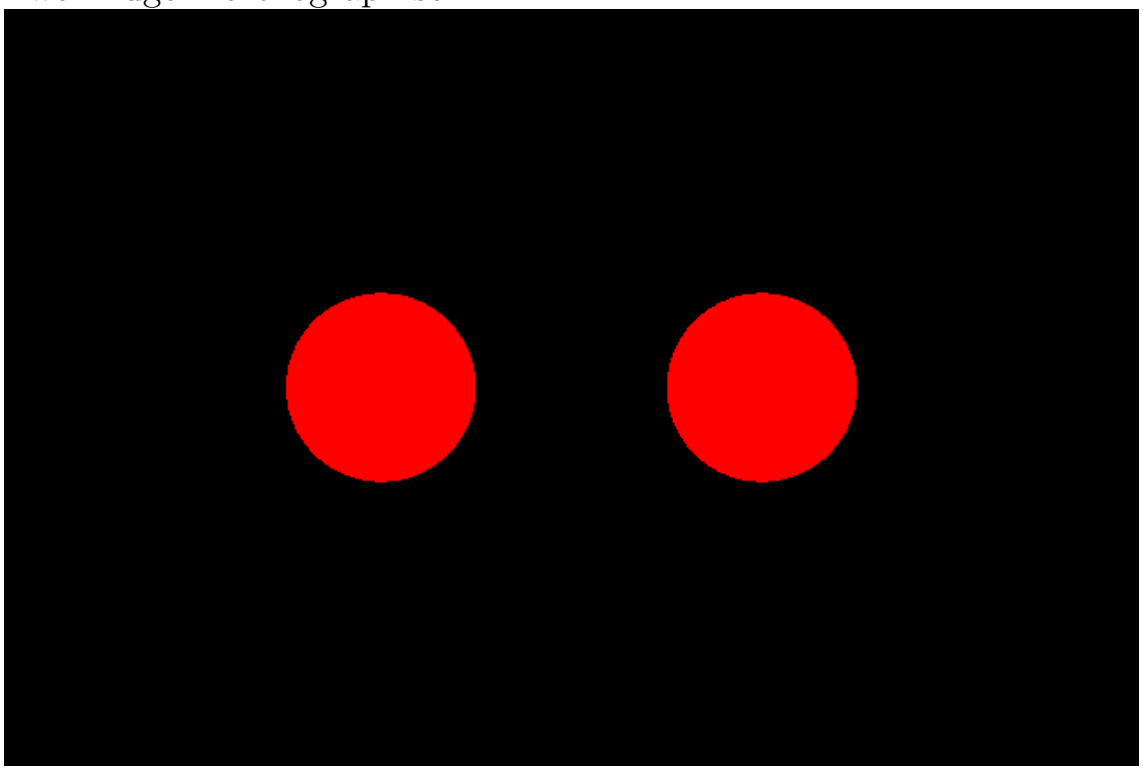
5.1.5 Abbildung 9

Zwei Kugeln perspektivisch.



5.1.6 Abbildung 10

Zwei Kugeln orthographisch.



6 Quellen